УДК 681.3.02 DOI: 10.31114/2078-7707-2022-3-40-44

О математических моделях цифровых микроэлектронных систем и проверке последовательности выполняемых функций на этапе проектирования

А.Д. Иванников

Институт проблем проектирования в микроэлектронике РАН, г. Москва

adi@ippm.ru

Аннотация — **Метод моделирования цифровых микроэлектронных систем широко используется при проектировании последних. При этом важным этапом является проверка проекта на правильность функционирования цифровой системы. На компьютерную модель проекта подаются проверочные тесты, и разработчик анализирует правильность или ошибочность изменения выходных и внутренних сигналов проектируемой системы. При этом должна осуществляться проверка правильности как выполнения всех функций системы, так и всех возможных последовательностей функций. В данной работе предлагается модель и методика проверки последовательностей функций цифровой системы.**

Ключевые слова — **моделирование цифровых систем, проверка последовательности выполняемых функций, отладка проектов на этапе проектирования.**

I. Введение

Еще в 90-ые году во многих работах указывалось на перспективность использования информационных технологий в различных областях современной человеческой деятельности: коммуникациях, образовании, науке и технике. В последующие тридцать лет информационные технологии постоянно развивались и находили применение во всех областях человеческой деятельности [1-4].

При проектировании цифровых систем управления объектами для отладки проектов широко используется метод моделирования [5-7]. На компьютерную модель цифровой системы подаются некоторые входные воздействия, а реакция модели проектируемой системы проверяется на соответствие техническому заданию. В связи с тем, что выполнение какой-либо функции может быть инициировано не только входным сигналом цифровой системы управления, но и самой цифровой системой, в качестве аргументов функционирования цифровых систем могут рассматриваться входные взаимодействия: последовательности сигналов, включающие как входные сигналы цифровой системы управления, так и ее выходные сигналы управления обменом [8]. При отладке методом моделирования важной задачей является выбор конечного числа конечных по времени тестовых входных взаимодействий (тестовых примеров) [9, 10].

В работе [8] показано, что функционирование цифровых систем управления объектами может быть представлена как последовательность выполняемых функций, каждая из которых задается подачей на цифровую систему некоторого входного взаимодействия. Иными словами каждая цифровая система выполняет некоторую последовательность функций из конечного алфавита **K**, причем выполнение каждой функции вызывается одним из входных взаимодействий определенного класса. Входные взаимодействия, задающие выполнение цифровой системой функции $k \in \mathbf{K}$, могут различаться значениями данных, временными соотношениями между отдельными сигналами в допустимых пределах, могут иметь и другие различия.

Отладочные тестовые примеры должны осуществлять как проверку правильности выполнения цифровой системой всех ее функций из множества **K**, так и правильности выполнения допустимых последовательностей функций $f \in \mathbf{F}$, где **F** – множество допустимых последовательностей функций из множества **K**. Алгоритм формирования минимального полного набора отладочных тестов для проверки правильности выполнения функций множества **K** предложен и проанализирован в [8]. Задачей данной работы является разработка метода формирования отладочного набора тестов для проверки правильности выполнения допустимых последовательностей функций.

II. Формальное представление множества последовательностей выполняемых функций

Рассматривая последовательное выполнение функций k_i и k_j как операцию умножения ·, а последовательность $k_i k_j$ как произведение, можно сказать, что на множестве последовательностей выполняемых функций **F** определена полугруппа, в общем случае частичная.

Рассмотрим полугруппу <**F**, ·>, в общем случае частичную, с конечным множеством **K** порождающих элементов *k*.

При этом возможно три случая.

1. Произведение $f' \cdot f''^{\,\prime\prime\prime\prime}$ определено для всех $f' \in \boldsymbol{F}, f'' \in \boldsymbol{F}$. В этом случае полугруппа <**F**, ·> является полной.

2. Произведение $f' \cdot f''^{''}$ определено, если f' и $f''^{''}$ могут быть представлены как $f' = k_{i_1} \cdot k_{i_2} \cdot \ldots \cdot k'$, $f'' = k'' \cdot k_{j_2} \cdot \ldots \cdot k_{j_l}$, причем произведение $'k' \cdot k''$ существует. В этом случае вопрос о существовании $f' \cdot f''^{''}$ сводится к вопросу о существовании произведений $k' \cdot k''$, где $k' \in \boldsymbol{K}, k'' \in \boldsymbol{K}$.

3. Произведение $f' \cdot f''^{''}$ определено, если существуют произведения $k_{i_1} \cdot k_{i_2} \cdot \ldots \cdot k' \cdot k''$, $k_{i_1} \cdot k_{i_2} \cdot \ldots \cdot k' \cdot k'' \cdot k_{j_2}$, …, $k_{i_1} \cdot k_{i_2} \cdot \ldots \cdot k' \cdot k'' \cdot k_{j_2} \cdot \ldots \cdot k_{j_l}$. В этом случае вопрос о существовании $f' \cdot f''^{''}$ сводится к вопросу о существовании произведения $f \cdot k$, где $f \in \mathbf{F}$, $k \in \mathbf{K}$.

Третий случай является наиболее общим, в связи с чем проведем его анализ.

Проектируемые цифровые системы всегда имеют конечное множество внутренних состояний. В связи с этим все существующие произведения могут быть заданы конечным числом правил.

Множество **F** есть множество слов в конечном алфавите **K**, которое может быть задано некоторой праволинейной грамматикой с конечным числом правил вывода. Множество допустимых слов **F** может быть задано конечным инициальным автоматом без выходов

$$A=(\mathbf{K},\mathbf{X},x_н,\varphi), \quad (1)$$

где **K** – множество входных символов;

X – множество состояний автомата А;

$x_н$ – начальное состояние, $x_н \in \mathbf{X}$;

φ: $\mathbf{K} \times \mathbf{X} \rightarrow \mathbf{X}$ – частичное переходное отображение.

Слова множества **F** могут заканчиваться любым символом из **K**. В связи с этим любое состояние из **X** может являться заключительным. Каждому состоянию x, $x \in \mathbf{X}$ соответствует подмножество $\mathbf{F}^x$, $\mathbf{F}^x \subset \mathbf{F}$, слов, заканчивающихся в состоянии x.

Автомат А задает все возможные последовательности функций, выполняемые цифровой системой, то есть множество допустимых слов **F**. Будем называть автомат А автоматом функций. Графическое задание автомата **A** в виде диаграммы переходов является наиболее наглядным и простым для использования разработчиками цифровой аппаратуры. Спецификации на входные взаимодействия цифровой могут определяться заданием множества входных взаимодействий, соответствующих каждой функции k, $k \in \mathbf{K}$, и автомата функций А.

III. Анализ последовательностей выполняемых функций

После составления алфавита функций **K**, выполняемых цифровой системой, необходимо рассмотреть всевозможные пары k_i, k_j с точки зрения существования их произведений. При этом желательно задать существование произведений $k_i \cdot k_j$ независимо от предыдущих взаимодействий. Такая попытка опирается на опыт проектирования, который показывает, что для многих цифровых систем или блоков это справедливо. Кроме того, при надежном проектировании системы должна быть предусмотрена ее реакция на любую последовательность входных взаимодействий. В случае последовательной подачи двух входных взаимодействий k_i, k_j, физически не допустимых или оставляющих внутреннее состояние не определенным, должна быть предусмотрена реакция цифровой системы, указывающая на ошибочность (недопустимость) ситуации. В этом случае произведение $k_i \cdot k_j$ существует и учитывается в спецификации. При разработке цифровой системы нет необходимости предусматривать реакцию на какую-то последовательность входных взаимодействий только в том случае, если инициация того или другого входного взаимодействия осуществляется самой цифровой системой или блоком.

При анализе всех возможных произведений $k_i \cdot k_j$ необходимо определить, существуют ли они независимо от предыстории. Наличие условия указывает на зависимость существования произведений от предыдущих входных взаимодействий.

В случае отсутствия условий для всех произведений имеем второй случай, то есть существование всех $k_i \cdot k_j$ не зависит от предыдущих взаимодействий. Тогда для каждого k_i, $k_i \in \mathbf{K}$ существует подмножество $\mathbf{K}^i$, $\mathbf{K}^i \subseteq \mathbf{K}$ такое, что $k_i \cdot k_j$, где $k_j \in \mathbf{K}^i$, всегда существует. В этом случае на множестве слов **F** задано отношение эквивалентности о, такое, что $(f_1,f_2) \in$ о тогда и только тогда, когда f_1 и f_2 кончаются на одну букву.

Отношение эквивалентности о определяет разбиение **F** на классы эквивалентности $\mathbf{F}^i$, i=1,2,…,n, где n – мощность алфавита **K**. При отождествлении классов $\mathbf{F}^i$, i=1,2,…,n и $\mathbf{F}^x$, $x \in \mathbf{X}$ и соответственно множества внутренних состояний автомата (1) и множества **K** автомат А может быть заменен автоматом [11]:

$$A'=(\mathbf{K},\mathbf{K},x_н,\varphi), \quad (2)$$

где **K** – множество входных символов и множество внутренних состояний;

$x_н$ – начальное состояние, $x_н \in \mathbf{K}$;

φ:$\mathbf{K} \times \mathbf{K} \rightarrow \mathbf{K}$ – частично определенное переходное отображение.

Каждое внутреннее состояние k, $k \in \mathbf{K}$ задает множество слов, заканчивающихся на букву k. Представление автомата функций в виде (2) весьма удобно как при составлении спецификаций на входные взаимодействия, так и при использовании для выбора отладочных тестов. В связи с этим необходимо скорректировать алфавит функций цифровой системы путем разделения элементов множества **K**, для которых имеются условия существования произведения $k_i \cdot k_j$, на несколько подэлементов.

Разделение элементов k_i на подэлементы и формирование нового алфавита **K** всегда может быть осуществлено в соответствии с различными внутренними состояниями автомата функций А. При этом всегда существует физический смысл в таком разделении, связанный с различными внутренними состояниями цифровой системы или ее части [12, 13].

IV. Методика формирования автомата функций

Автомат $\mathbf{A}'$ является неприведенным. С целью упрощения выделим в $\mathbf{A}'$ классы эквивалентных состояний и заменим каждый класс на одно состояние. Применительно к автомату вида (2) приведение означает выделение таких подмножеств $\widehat{\mathbf{K}}$ состояний, $\widehat{\mathbf{K}} \subset \mathbf{K}$, что для всех $\hat{k}$, $\hat{k} \in \widehat{\mathbf{K}}$, отображение $\varphi(k,\hat{k})$ определено для одинакового подмножества $\widetilde{\mathbf{K}}$ входных символов, и замену подмножеств $\widehat{\mathbf{K}}$ одним состоянием. Выделение подмножеств $\widehat{\mathbf{K}}$ может быть осуществлено путем поиска одинаковых наборов функций в графе функций. После приведения автомата $\mathbf{A}'$ получим автомат функций

$$\mathbf{A}''=(\mathbf{K},\mathfrak{R},x_н,\varphi:\mathbf{K}\times\mathfrak{R}\to\mathfrak{R}),$$

где $\mathbf{K}$ – алфавит функций;

$\mathfrak{R}$ - множество подмножеств $\mathbf{K}$, причем $\mathbf{K}=\cup_i \overline{K}$, $\overline{\mathbf{K}}_{i_1}\cap\overline{\mathbf{K}}_{i_2}=\varnothing$ при $\overline{\mathbf{K}}_{i_1}\neq\overline{\mathbf{K}}_{i_2}$; $\mathfrak{R}$ - есть множество состояний приведенного автомата;

$x_н$ – начальное состояние, $x_н \in \mathfrak{R}$.

Для дальнейшего упрощения автомата $\mathbf{A}''$ можно вновь полностью или частично объединить функции k – символы входного алфавита, соответствующие подаче одних и тех же входных взаимодействий в различных состояниях цифровой системы. Кроме того, можно объединить k_i и k_j – элементы входного алфавита, если все помеченные ими ребра в графе переходов автомата $\mathbf{A}''$ всегда начинаются и заканчиваются в одних и тех же вершинах. Однако, окончательное решение об объединении таких ребер принимает разработчик на основе физического смысла входных взаимодействий.

После проведения последней операции – объединения ряда ребер с соответствующим объединением элементов алфавита $\mathbf{K}$ – получим окончательный вид автомата функций:

$$\mathbf{A}_\phi=(\mathbf{K},\mathbf{X},x_н,\varphi:\mathbf{K}\times\mathbf{X}\to\mathbf{X}), \quad (3)$$

где $\mathbf{K}$ – модифицированный алфавит входных символов – функций цифровой системы;

$\mathbf{X}$ –множество внутренних состояний;

$x_н$ – начальное состояние, $x_н \in \mathbf{X}$;

φ - частично определенное переходное отображение.

Автомат функций вида (3) вместе с множествами $\mathbf{M}^k$ входных взаимодействий, соответствующих каждой функции k, является формальной спецификацией входных взаимодействий $\mathbf{M}$. При этом задание множества входных взаимодействий в виде $\mathbf{M}=(\mathbf{F},\{\mathbf{M}^k\})$ заменяется следующим образом:

$$\mathbf{M}=(\mathbf{K},\mathbf{A}_\phi,\{\mathbf{M}^k \mid k\in\mathbf{K}\}), \quad (4)$$

где $\mathbf{K}$ – алфавит функций;

$\mathbf{A}_\phi$ – автомат функций вида (3), определяющий допустимые последовательности функций $\mathbf{F}$;

$\mathbf{M}^k$ – множество входных взаимодействий, соответствующих k-ой функции цифровой системы или блока.

Методика получения автомата функций по неформальному описанию работы цифровой системы или блока, приводимому в ТЗ, состоит в следующем.

1. Выделить перечень функций и соответствующие входные взаимодействия.
2. Составить таблицу возможных последовательных пар функций с указанием условий существования таких пар.
3. Разбить элементы перечня функций на подфункции либо по физическому смыслу, либо в зависимости от состояния системы, в котором инициируется второе входное взаимодействие пары, таким образом, чтобы можно было задать возможные пары функций независимо от предыстории работы.
4. Объединить эквивалентные состояния полученного автомата функций.
5. Объединить функции, разделенные на этапе 3, и возможно, другие функции, всегда параллельные на графе переходов автомата функций.

Следует отметить, что выделение набора функций цифровой системы в определенной степени процесс субъективный. Однако, несмотря на различный вид автомата функций $\mathbf{A}_\phi$ при выборе различный множеств $\mathbf{K}$, в связи с однозначностью требований к поведению цифровой системы или блока получаемая спецификация входных взаимодействий приведет к проверке функционирования отлаживаемой системы одинаковым образом.

V. Предлагаемый подход к тестированию последовательностей выполнения функций цифровых систем

Для задания отладочных тестов в спроектированной цифровой системе необходимо выделить классы внутренних состояний $[x_i]$, гомоморфные каждому внутреннему состоянию x_i автомата функций $\mathbf{A}_\phi$. Множество отладочных тестов должно быть составлено таким образом, чтобы для каждого x_i, $x_i \in \mathbf{X}$ автомата $\mathbf{A}_\phi$ предусматривалась поочередная подача входных взаимодействий, соответствующих функциям ребер, выходящих из x_i на графе переходов автомата $\mathbf{A}_\phi$. При этом в процессе моделирования необходимо убедиться, что при установке автомата $\mathbf{A}_\mathbf{M}$ в любое состояние из $[x_i]$ и подаче отладочного теста k_j автомат $\mathbf{A}_\mathbf{M}$ переходит в одно из состояний $[\varphi(k_j, x_i)]$ и выдает правильное выходное воздействие. Такой методы выбора отладочных тестов соответствует тестированию всех переходов автомата функций спецификации.

Литература

[1] Тихонов А.Н., Иванников А.Д. Информатизация российского образования и общества в целом // Международное сотрудничество. 1997. № 4. С. 1.

[2] Климов А.В., Левченко Н.Н., Окунев А.С., Стемпковский А.Л. Методы адаптации параллельной потоковой вычислительной системы под задачи отдельных классов

// Информационные технологии и вычислительные системы. 2009. №3. С. 12-21.
[3] Иванников А.Д., Тихонов А.Н., Цветков В.Я. Критерии готовности к использованию информационных технологий // Международный журнал прикладных и фундаментальных исследований. 2009. № 3. С. 84-85.
[4] Иванников А.Д. Тематические интернет-порталы как средство агрегации электронного контента в заданной предметной области // Информационные технологии. 2014. №3. С. 43-48.
[5] Lin, Yi-Li; Su, Alvin W.Y. Functional Verification for SoC Software/Hardware Co-Design: From Virtual Platform to Physical Platform // 2011 IEEE International SOC Conference (SOCC), pp. 201-206.
[6] Shi, Jin; Liu, Weichao; Jiang, Ming; el al. Software Hardware Co-Simulation and Co-Verification in Safety Critical System Design // 2013 IEEE International Conference on Intelligent Rail Transportation (ICIRT), pp. 71-74.
[7] Кащеев Н.И., Пономарев Д.М., Подъяблонский Ф.М. Построение тестов цифровых схем с использованием обобщенной модели неисправностей и непрерывного подхода к моделированию // Вестник Нижегородского университета им. Н.И.Лобачевского. 2011. №3 (2). С. 72-77.
[8] Иванников А.Д. Формирование отладочного набора тестов для проверки функций цифровых систем управления объектами // Мехатроника, автоматизация, управление. 2017. Т. 18. № 12. С. 795-801.
[9] Nguen M.D. Hardware/software formal co-verification using hardware verification techniques // Fourth Int. Conf. on Communications and Electronics (ICCE). 2012. Pp. 465-470.
[10] Стемпковский А.Л., Тельпухов Д.В., Соловьев Р.А., Соловьев А.Н., Мячиков М.В. Моделирование возникновения неисправностей для оценки надежностных характеристик логических схем // Иформационные технологии. 2014. № 11. С. 30-36.
[11] Оллонгрен А. Определение языков программирования интерпретирующими автоматами. М.: Мир, 1977. 288 с.
[12] Romanov A.Yu., Ivannikov A.D., Romanova I.I. Simulation and synthesis of network-on-chip by using NOCSIPM HDL library. 2016 IEEE 36th International Conference on Electronics and Nanotechnology, ELNANO 2016 – Conference Proceedings. 36. 2016. Pp. 300-303.
[13] Гаврилов С.В., Иванова Г.А., Стемпковский А.Л. Теоретико-графовая модель сложно-функциональных блоков для КМОП технологий с трехмерной структурой транзистора // Известия ЮФУ. Технические науки. 2014. № 7 (156). С. 58-68.

On Mathematical Models of Digital Microelectronic Systems and Verification of the Sequence of Functions Performed at the Design Stage

A.D. Ivannikov

Institute for design problems in microelectronics of RAS, Moscow, adi@ippm.ru

Abstract — **The method of digital microelectronic system simulation is widely used in the design of the latter. At the same time, an important step is to check the design for the correct functioning of the digital system. Verification tests are submitted to the computer model of the design, and the developer analyzes the correctness or error of changing the output and internal signals of the system being designed. In this case, the correctness of both the performance of all system functions and all possible sequences of functions should be checked. This paper proposes a model and method for checking the sequences of functions of a digital system design.**

Keywords — **digital system simulation, fulfilled function sequence checking, design debugging on early stage.**

REFERENCES

[1] Tikhonov A.N., Ivannikov A.D. Informatization of Russian education and society as a whole // International cooperation. 1997. No. 4. P. 1.
[2] Klimov A.V., Levchenko N.N., Okunev A.S., Stemokovskiy A.L. Parallel dataflow computing system adaptation for specific task classes // Journal of Information Technologies and Computing Systems. 2009. No. 3. Pp. 12-21.
[3] Ivannikov A.D., Tikhonov A.N., Cvetkov V.Ya. Readiness criteria for information technologies usage //International Journal of Applied and Fundamental Research. 2009. No. 3. Pp. 84-85.
[4] Ivannikov A.D. Subject Internet portals as the means of aggregating electronic content in a given subject area // Information technologies. 2014. No. 3. Pp. 43-48.
[5] Lin, Yi-Li; Su, Alvin W.Y. Functional Verification for SoC Software/Hardware Co-Design: From Virtual Platform to Physical Platform // 2011 IEEE International SOC Conference (SOCC), pp. 201-206.
[6] Shi, Jin; Liu, Weichao; Jiang, Ming; el al. Software Hardware Co-Simulation and Co-Verification in Safety Critical System Design // 2013 IEEE International Conference on Intelligent Rail Transportation (ICIRT), pp. 71-74.
[7] Kasheev N. I., Ponomarev D. M., Podyablonsky F. M. Postroenie testov cifrovih chem s ispolsovaniem obobshennoy modeli neispravnostei i neprerivnogo podhoda k modelirovaniu (Digital circuits tests generation based on generalized malfunction model and continuous simulation approach), Vestnik Nijegorodskogo Universiteta, 2011, no. 3 (2), pp. 72—77.
[8] Ivannikov A.D. Debugging Input Set Generation for Testing of Control Digital Systems Functions // Mekhatronika, Avtomatizatciya, Upravlenie. 2017. Vol. 18. No.12. Pp. 795-801.
[9] Nguen M.D. Hardware/software formal co-verification using hardware verification techniques // Fourth Int. Conf. on Communications and Electronics (ICCE). 2012. Pp. 465-470.

[10] Stempkovskiy A.L., Telpukhov D.V., Solovyev R.A., Solovyev A.N., Myachikov M.V. Fault simulation technique for ligic ciecuits reliability characteristics evaluation // Informacionnie Technologii. 2014. No. 11. Pp. 30-36.

[11] Ollongren A. Opredelenie yazikiv programmirovaniya interpretiruushimi avtomatami (Programming Languages Definition by Interpretation Automata). Moscow. Mir. 1977. 288 p.

[12] Romanov A.Yu., Ivannikov A.D., Romanova I.I. Simulation and synthesis of network-on-chip by using NOCSIPM HDL library. 2016 IEEE 36th International Conference on Electronics and Nanotechnology, ELNANO 2016 – Conference Proceedings. 36. 2016. Pp. 300-303.

[13] Gavriliv S.V., Ivanova G.A., Stempkovsky A.L. Theoretical Graph Model of Complex Functional Blocks for CMOS Technology with Three Dimension Transistor Structure. Izvestiya UFU. Tehnicheskie Nauki. 2014. No. 7 (156). Pp.58-68.